# Web Scraping with Python

Let’s say you need to scrape a lot of information from the web, and time is of the essence. What alternative would there be to manually accessing each website and retrieving the information? “Web Scraping” is the technique that can be used.

Web scraping merely facilitates and accelerates the process. This document shows how to extract data from a website with a demonstration. The following topics that are covered in this document : ♣

* Why is Web Scraping Used? ♣
* What Is Web Scraping? ♣
* Is Web Scraping Legal? ♣
* Why is Python Good For Web Scraping? ♣
* How Do You Scrape Data From A Website? ♣
* Libraries used for Web Scraping ♣
* An example of scraping the APH shop.

## Why is Web Scraping Used?

Web scraping is used to collect information from websites. But why does someone have to collect data from websites? To know about this, let’s look at the applications of web scraping: • Price Comparison: some services use web scraping to collect data from online shopping websites and use it to compare the prices of products. • Email address gathering: Many companies that use email as a medium for marketing use web scraping to collect email ID and then send bulk emails. • Social Media Scraping: Web scraping is used to collect data from Social Media websites such as Twitter to find out what’s trending. • Research and Development: Web scraping is used to collect a large set of data (Statistics, General Information, Temperature, etc.) from websites, which are analyzed and used to carry out Surveys or for R&D. • Job listings: Details regarding job openings, interviews are collected from different websites and then listed in one place so that it is easily accessible to the user.

## What is Web Scraping?

Web scraping is one of the automated processes for gathering extensive information from the World Wide Web. The information found on the websites is disorganized. To shop this data in a more organized fashion, web scraping is a useful tool. Online services, application programming interfaces (APIs), and custom code are just some of the options for scraping websites.

## Is Web Scraping Legal?

Some websites allow web scraping, and some don’t. To know whether a website allows web scraping or not, look at the website’s “robots.txt” file. Find this file by appending “/robots.txt” to the URL being scraped . For this example, the site being scraped is the APH web shop . So, to see the “robots.txt” file, the URL is [www.aph.org/robots.txt](http://www.aph.org/robots.txt).

## is Python Good for Web Scraping?

Here is the list of features of Python which makes it suitable for web scraping.

* Ease of Use: Python programming is simple to code. You do not have to add semi-colons “;” or curly braces “{}” anywhere. This makes it less messy and easy to use.
* Large Collection of Libraries: Python has a vast collection of libraries such as numpy, Matlplotlib, Pandas etc., which provides methods and services for various purposes. Hence, it is suitable for web scraping and for further manipulation of extracted data.
* Dynamically typed: In Python, you don’t have to define datatypes for variables, you can directly use the variables wherever required. This saves time and makes your job faster.
* Easily Understandable Syntax: Python syntax is easily understandable mainly because reading Python code is similar to reading a statement in English. It is expressive and easily readable, and the indentation used in Python also helps the user to differentiate between different scope/blocks in the code.
* Small code, large task: Web scraping is used to save time. But what’s the use if you spend more time writing the code? Well, you don’t have to. In Python, you can write small codes to do large tasks. Hence, you save time even while writing the code.
* Community: What if you get stuck while writing the code? You don’t have to worry. Python community is one of the biggest and most active communities, where you can seek help from.

## How Do You Scrape Data From A Website?

When the code is run for web scraping, a request is sent to the URL set in the code . As a response to the request, the server sends the HTML or XML page. The code then parses the HTML or XML page, finds the data and extracts it. To extract data using web scraping with python, Follow these basic steps:

1. Find the URL that you want to scrape

2. Inspect the Page

3. Find the data you want to extract

4. Write the code

5. Run the code and extract the data

## Scraping APH web shop Pre-requisites

For this example, python 3 is needed with the following modules installed.

* Requests
* BeautifulSoup

The page could be scraped by using just strings and slicing with string functions, but it would be more difficult than it is when using BeautifulSoup.

## Step 1 Find the URL that you want to scrape.

To get the stock of the Mantis braille display. This is the page to be scraped.

https://www.aph.org/product/mantis-q40/

## Step 2: Inspecting the Page

To inspect the page, right click on the stock element and click on “Inspect”. When you click on the “Inspect” tab, you will see a “Browser Inspector Box” open. Go ahead and click on the in stock region.

## Step 3: Find the data.

In this case what is in the inspect field is:

<p class=”stock in-stock”>1### instock</p>

The number and text may have changed by the time you try this.

Note this information for the next step.

## Step 4: Write the code.

Open a new Python file in your editor call it something like instock.py.

Add the following:

#load the requests module that is used to get the page

import requests.

#load the beautiful soup module for parsing the page and getting info

from bs4 import BeautifulSoup

# save the HTML reference of the web site.

URL = <https://www.aph.org/product/mantis-q40/>

#Request the page from the web and store it in variable named page.

page = requests.get(URL)

# parse the page with Beautiful soup and save it in the variable soup

soup = BeautifulSoup(page.content, "html.parser")

# find the tag that you found with inspect using find all. From beautiful soup

# Remember the tag was <P> and has the class of “stock in-stock”

in\_stock = soup.find\_all("p", class\_="stock")   [0].get\_text()

#print the information you retrieved.

print(f"Mantis: {in\_stock}")

That ends the code. Now run it by doing:

Python instock.py

That is all there is to it. From that you can make larger more powerful web scraping by changing the website and product or just searching for something else like the weather on weather.com. Always remember to check robots.txt and make sure the companies are OK with you scraping their page.